MEGA – UE4

For **parts 1-3** of this exercise you will use `Python 3.9` and the library `pyvista`.

For **part 4**, you will need MeVisLab (https://www.mevislab.de/download, see also UE3).

Part 1: Marching cubes

The script `marching_cubes_task1.py` is a simple implementation that allows you to load a data set and extract an isosurface from it using the marching cubes algorithm. Use the application to load the `shoulder.vti` data set that we provided you on TUWEL. Then, you need to select the adequate scalar value for the variable `isovalue`, in order to extract a meaningful structure of the bone.

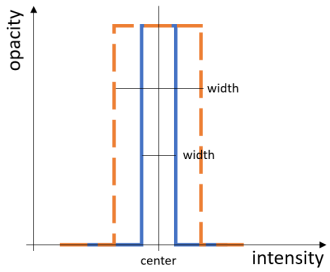
Read the documentation of the `add_mesh` function (link) and focus particularly on the parameters `color` and `opacity`. Change the values of these two parameters in your code to understand what they do. Then, look at the parameters `ambient`, `diffuse`, `specular`. Set them to 1,0,0 respectively. What is happening to your rendering? Now, set them to 0,1,0 and then to 0,0,1. Finally, check what does a 1,1,0 combination do. Explain what happens in each of these cases. Find a good configuration for these parameter values. Support your findings with screenshots.

Part 2: Volume rendering

The script `volume_task2.py` is another implementation that allows you to load a data set and create a direct volume rendering from the it. Use the application to load the `shoulder.vti` data set that we provided you on TUWEL. Now, go ahead and change the colors (`cmap`), the `opacity`, the shading (`shade`), and the `blending` parameters of the `add_volume` function (link to documentation). Explain and show with correspondingly generated images what these parameters do. Select a final rendering that adequately represents the bones within the volume. Support your findings with screenshots.

Part 3: Creating your own transfer function with two channels (center/spread)

The script `volume_task3.py` allows you to further manipulate the transfer function. There are two sliders now in the UI: the spread (in class we referred to this as "*width*") and the center, which control the corresponding parameters. The `spread` parameter determines how big is the neighborhood of opaque densities around the `center` density. An example of two cases with the same center but different spread (i.e. width) is shown below.



Now, the function `transfer_function` in your script does not work as intended. Modify this transfer function, so *the densities around the center density are more opaque*. Support your findings with screenshots.

Part 4: A simple volume rendering in MeVisLab

1. Load the `shoulder.vti`
2. Build a MeVisLab basic network with the following modules: **SoExaminerViewer**, **SoGVRVolumeRenderer, SoSeparator** and **SoLUTEditor**. Notice that **SoGVRVolumeRenderer** uses the capabilities of the graphics cards, so it depends on your graphics card. *Hint: If you have trouble constructing your network, try right clicking on a module > Help > Show Example Network.*
3. Change the 1D transfer functions and come up with one transfer function that visualizes meaningful volume information.
4. Look at the options of the volume rendering and generate different versions of it using "Illuminated", "Direct", "Blend (compositing)", "MIP" options. What do these do?

Support your findings with screenshots.

For all cases, you should deliver your code (for parts 1-3) and your network (part 4), as well as your answers with adequate screenshots (where needed to explain your findings).